# Books

**Software Reuse: Architecture, Process and Organization for Business Success**, Ivar Jacobson, Martin Griss, and Patrik Jonsson, Addison-Wesley Publishing Co., Reading, MA, 1997. 497 pp. (ISBN 0-201-92476-5).

Software reuse is an approach to development that we all cite as a best practice but don't really understand how to do. We know its potential for improving quality and productivity, but there are few large-scale examples to show us how good reuse should work as a long-term strategy. Often, we institute a trial reuse program and hold our collective breath, hoping that we will see some benefit and wishing that reuse becomes common, well-accepted practice.

Well, now we can breathe more easily. In their book, *Software Reuse: Architecture, Process and Organization for Business Success*, Jacobson, Griss, and Jonsson take a broad, business view, showing us what is necessary to make reuse successful. They have analyzed reuse programs good and bad, short- and long-term, large and small. By combining good business sense with an understanding of architecture and technology, they make a series of large- and fine-grained recommendations that can help each of us to understand not only what makes reuse work but why.

The authors begin by explaining that reuse is not just a technology; it is a business practice, and it must be embedded in the larger business processes that your company or organization enacts. They debunk some of the more grandiose claims found in the literature, and propose instead that reuse be adopted in small steps, each of which builds on the previous ones:

- Informal code reuse to reduce development time
- Black-box code reuse to reduce maintenance costs
- Managed work product reuse to broaden coverage
- Architected reuse to achieve interoperability and high reuse levels
- Domain-specific reuse to achieve rapid custom product development

Each step requires additional investment, experience, and time, but yields larger benefits in terms of time-to-market, cost, and quality.

The authors distill a great deal of experience into several rules of thumb. For example, they list several guidelines about reuse economics (p. 23):

- A component has to be used three to five times in application projects to recover the initial cost of creating it and the ongoing cost of supporting it.
- It costs 1.5 to 3 times as much to create and support a reusable component as it does to implement a similar component for a single application.
- It costs only one-quarter as much to use a reusable component as to develop a new one from scratch.
- It takes two or three product cycles—usually about three years—before the benefits of reuse become significant.

Similarly, they give us guidance on organizational principles, such as (p. 27):

- Maintain top-management leadership and financial backing over the long term.
- Plan and adapt the system architecture, the development processes, and the organization to the necessities of reuse in a systematic but incremental fashion. Start with small pilot projects and then scale up.
- Move to an explicitly managed reuse organization that separates the creation of reusable components

©Copyright 1998 by International Business Machines Corporation.


0018-8670/98/$5.00 © 1998 IBM

from their reuse in application systems and provides an explicit support function.

Unlike other reuse books and papers whose guidelines focus on similarity, the ones in this book target understanding the variance among components. "If we made no provision for variance, we would have to resign ourselves to the burden of a very large number of 'concrete' component systems" (p. 39). Consequently, Jacobson, Griss, and Jonsson describe a reuse business composed of application family engineering, application system engineering, component system engineering, and managing the reuse business. Each of these major activities is discussed in great detail, acknowledging that each is done by different participants with very different audiences and goals.

For example, architectural issues are discussed in terms of variability mechanisms that allow common functionality and structure to be expanded in a carefully controlled way. Chapter 5 tells us about variation points and "extends" constructs that allow us to design families of similar components. Object-orientation, while not a necessary condition for reuse, is used to provide a vocabulary for discussing notions of reusable architectures.

The chapters on business process are particularly useful. Jacobson, Griss, and Jonsson explain how to do business process engineering and how to organize for reuse. They emphasize that you must understand your company's business and its needs, purpose, players, structures, and problems before you can begin to think about software-based solutions.

The authors also discuss the important role of software measurement in understanding reusability and in tracking the effects of reuse in the software and in your business. They suggest that software reuse measurement is useful only when you first take the time to understand the underlying processes and business climate. And they emphasize that a great deal of up-front investment is necessary for reaping long-term benefits.

Throughout the book, the authors use a common banking example to illustrate their concepts. Using this example, they show how a family of similar applications can be designed for reuse, and how a risk analysis is necessary for deciding among several technology options. In addition, Jacobson, Griss, and Jonsson cite examples from their own experiences at Ericsson, Hewlett-Packard, and Rational Software, as well as from mainstream developers such as Microsoft and ParcPlace to make their points. Each chapter ends with suggestions for additional reading; these references point not only to the software literature but also to the business literature that deals with successful technology transfer.

There is a wealth of information in the book for all of the players involved in reuse: requirements analysts, designers, programmers, testers, managers, and business strategists. Each chapter tackles a different aspect of reuse, and all blend together to provide the most comprehensive and useful picture of reuse that I have found. If you are thinking about starting a reuse program, this book is essential reading for helping you to understand the important variables in your decision-making process. If you are working to understand or improve your existing reuse efforts, this book helps to make more visible the key elements that lead to reuse success. And if you are recovering from a failed reuse effort, this book is useful for analyzing what went wrong and why you should try again.

Shari Lawrence Pfleeger
Systems/Software, Inc.
Washington, D.C.

Note—The books reviewed are those the Editor thinks might be of interest to our readers. The reviews express the opinions of the reviewers.